Sip Client in Java

The Agent class is the main entity.

Ozonetel communicates with the Agent using SIP+SDP and sets up an RTP session between the two. The media packets received by the Agent are then forwarded to the VoiceBot. Once the processing is done on the VoiceBot's end, it sends the processed packets via websocket to the Agent which again communicates it back to Ozonetel using the RTP session.

Agent

The Agent class is the main class. It spawns three threads, one for the SipExtension, one for the DataSender which sends data to Ozonetel and one for the DataReceiver which receives data from Ozonetel. Apart from this the websocket is started which internally spawns two threads. The websocket communicates with the VoiceBot , forwarding the Ozonetel data packets and receiving the processed packets to be sent to Ozonetel.

AgentManager

This maintains a map of each Agent with their name. No separate map for the state/configuration is required since the Agent itself encapsulates its state and configuration. This is useful to track the current state of the Agent and also to retrieve any configuration.

AgentConfiguration

This encapsulates all the configuration fields which are required. It is currently being set by reading in from a yaml file but it could be created in a plethora of ways. It could be read in from a JSON body sent via a POST request or it could be read in from the application.properties file. You can design a different way to create the AgentConfiguration but ensure the getter methods remain the same to ensure code compatibility.

SipExtension

The SipExtension has to send only one request, the REGISTER request. Rest of the time it acts as a UAS, only dealing with requests sent from Ozonetel’s side. It has to only send back responses.
The REGISTER request handles Digest Authentication as well, a utility class is written for that in utils/DigestMD5Converter
It creates the request using sip/SipRequestCreator and keeps sending it in some regular intervals using a timer (to handle the expiry of the User Agent with the Registrar)

The remaining response creation flow is standard and in line with what is mentioned in the RFCs
Keep in mind that a REGISTER request does not create a Dialog.

NOTE: Understand what a Dialog and Transaction is

The SipExtension can send the REGISTER request anywhere, just mention it in the yaml config file (sipRegistrarIp, sipRegistrarPort). And it can receive requests from anyone, as long as the sender can retrieve the address and port of our SipExtension (the sipLocalIp and sipLocalPort).

It can send back responses to a proxy server or directly to another UA (thus acting in a peer-to-peer fashion). This does not have to be handle explicitly by us since we are only sending the responses.
Remember to set the SDP appropriately to convey the necessary information (media codecs, media ports etc)

DataSender
Any entity which implements this will keep polling from the outbound queue and send data back to Ozonetel. Currently, an RTP implementation is provided but one can easily incorporate and extend other protocols such as SRTP.

DataReceiver
Any entity which implements this will keep pushing data it receives from Ozonetel into the inbound queue. Currently, an RTP implementation is provided but one can easily incorporate and extend other protocols such as SRTP.

Websocket
The websocket is used for media transfer with the voicebot, It is connected with the Agent using the inbound and outbound queues. It continuously polls data from the inbound queue and sends it to the voicebot. The processed data which is to be sent back to Ozonetel is then received by this websocket entity and pushed into the outbound queue.

Assumptions

1. The data which is sent back by the websocket is properly formatted i.e. it is encapsulated in an RTP packet.

2. Each Agent in the current implementation will have a separate listening point i.e. a separate port .
   If only one port /few ports are to be used, extract the SipExtension from the Agent part and keep it global for the Agents. Then check the To-header of the requests received from Ozonetel and set the states and everything of the Agents accordingly.

Design Choices
Using ConcurrentLinkedQueue for the inbound and outbound queues, can choose a blocking queue to save memory cycles while polling from the queues. However, it might have some other drawbacks like more memory assignment or slower throughput.
Refer https://www.baeldung.com/java-concurrent-queues

Small details are mentioned in the code as comments. JavaDocs have also been added.